Лабораторная работа № 8
ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

Цель работы

Изучить применение электронной цифровой подписи, основанной на криптографическом хэше и асимметричном алгоритме для решения задач аутентификации, контроля целостности и подтверждения обязательств с использованием средств .NETSecurityFramework.

Методические указания

1. Функции хеширования.

Хеш – это функция, которая ставит в соответствие небольшой, фиксированного размера объем двоичных данных произвольному, сколь угодно большому объему входных данных.

$$H = H(M),$$

где M – прообраз - сообщение произвольной длины,

H(M) – значение функции хеширования фиксированной длины.

Значение функции хеширования называют также ***хеш-код, свертка, функция сжатия, профиль сообщения, дайджест сообщения, отпечаток пальца***.

Значение хэш-кода присоединяется к сообщению отправителем. Получатель устанавливает аутентичность сообщения путем повторного вычисления значения функции хеширования.Т.о. целью использования функции хеширования является получение «дактилоскопической» характеристики файла или любого блока данных.

Однако в общем случае однозначного соответствия между исходными данными и хеш-кодом нет в силу того, что количество значений хеш-функций меньше чем вариантов входного массива; существует множество массивов, дающих одинаковые хеш-коды — так называемые **коллизии.** Вероятность возникновения коллизий играет важную роль в оценке качества хеш-функций.

Существует множество алгоритмов хеширования с различными характеристиками (разрядность, вычислительная сложность, криптостойкостьи т.п.). Выбор той или иной хеш-функции определяется спецификой решаемой задачи.

Простые хеш-функции широко используются в программировании, не связанном с криптографией, например, для обнаружения ошибок или для быстрого поиска объектов по хеш-таблице. Виртуальный метод GetHashCodeобъекта Object – это пример простой хеш-функции, создающей 32-битовый хеш.

Криптографические хеш-функции должны обладать дополнительными свойствами, обеспечивающими их криптографическую стойкость и позволяющими применять их в криптографических приложениях.

Идеальная криптографическая хеш-функция должна обладать следующими свойствами:

1. Быть применимой к блоку данных любой длины.
2. Давать на выходе значение фиксированной длины.
3. Значение H(x) должно вычисляться относительно легко для любого заданного x, а алгоритм вычисления должен быть практичным с точки зрения аппаратной и программной реализации.

4. Изменение в прообразе даже одного бита должно приводить к изменению приблизительно половины битов в значении хеш-функции (диффузия или лавинный эффект).

5. Свойство однонаправленности.Однонаправленной хеш-функцией называют хеш-функцию для которой вычислить значение хеш-функции по прообразу несложно, а сгенерировать прообраз, который свернется к данной величине, очень трудно.

6. Для любого данного блока x должно быть практически невозможным вычислить y≠x, для которого H(x)=H(y). Такое свойство называется слабой сопротивляемостью коллизиям или стойкость к коллизиям первого рода.

7. Должно быть практически невозможно подобрать пару сообщений (M, M'), имеющих одинаковый хеш-код. Это свойство называется сильной сопротивляемости коллизиям или стойкость кколлизиям второго рода.

2. Классы, реализующие хеш-алгоритмы, поддерживаемые в .NETFramework

Перечисленные ниже классы, реализующие известные алгоритмы хеширования, являются производными абстрактного класса HashAlgorithm пространства имен System.Security.Cryptography. В таблице слева приводится имя абстрактного класса, а справа – имя производного конкретного класса.

Абстрактный класс	Производный класс
MD5	MD5CryptoServiceProvider
SHA1	SHA1Managed и SHA1CryptoServiceProvider
SHA256	SHA256Managed
SHA384	SHA384Managed
SHA512	SHA512Managed

Абстрактные классы невозможно использовать напрямую, создавая на их основе экземпляры объектов. Из каждого такого класса производятся конкретные классы реализации. Классы, имена которых заканчиваются на CryptoServiceProvider, реализованы с использованием интерфейса CryptoAPI, предоставляемого операционной системой. Классы, имена которых заканчиваются на Managed, реализованы полностью средствами контролируемого C#-кода, без использования CryptoAPI.

Класс HashAlgorithm обладает публичным свойством Hash, которое представляет собой байтовый массив, содержащий вычисленный хеш-код.

Публичное свойство HashSizeсодержит значение размера хеш-кода в битах.

Самый важный публичный метод класса HashAlgorithm – метод ComputeHash. Он возвращает значение хеш-кода в виде массива байтов, вычисленное для заданных во входном параметре входных данных.Входные данные также представляются в виде массива байтов.

В следующем примере иллюстрируется использование класса HashAlgorithmна примере конкретного производного класса, инкапсулирующего алгоритм SHA-1. Предполагается, что входные данные заданы в байтовом массиве messageByteArray.Значение хеш-кода возвращается в массив sha1Hash.

```
HashAlgorithmsha1 = newSHA1CryptoServiceProvider();
byte[] sha1Hash = sha1.ComputeHash(messageByteArray);
```

3. Идентификаторы объектов

Международный стандарт ASN.1 OID (ObjectIdentifiers) предназначен для формирования уникальных идентификаторов для компьютерных форматов, логически организованных в иерархию имен. Он поддерживается многими организациями, включая ANSI. Существует большое число идентификаторов OID, идентифицирующих конкретные протоколы, алгоритмы и форматы данных. В частности, уникальные идентификаторы OIDимеют многие криптографические алгоритмы, признанные ANSI.

Некоторые значения идентификаторов приведены в таблице.

Криптографический хеш-алгоритм	**OID**
MD5	1.2.840.113549.2.5
SHA-1	1.3.14.3.2.26
SHA-2	562.16.840.1.101.3.4.2.1
SHA-3	842.16.840.1.101.3.4.2.2
SHA-5	122.16.840.1.101.3.4.2.3

Идентификаторы OID необходимо использовать в определенных методах классов .NETSecurityFramework.

Например, в методах SignHash и VerifyHashклассов RSACryptoServiceProviderи DSACryptoServiceProvider.

В приведенном фрагменте кода показано, как идентификатор OID алгоритма хеширования используется в качестве параметра метода SingHashкласса RSACryptoServiceProvider.

RSACryptoServiceProviderrsa = new RSACryptoServiceProvider();

Signaturebytes = rsa.SignHash(hashbytes, “1.3.14.3.2.26”);

Сначала создается объект RSAс ключом по умолчанию. Затем с помощью методаSingHashхеш-код подписывается (шифруется)личным ключом. Параметрами метода являются: байтовый массив, содержащий сам хеш-код и идентификатор OID алгоритма хеширования SHA-1. Предполагается, что хеш-код, переменная hashbytes,уже создан вызовом метода ComputeHashкласса SHA1.

4. Электронная цифровая подпись

Аутентификация сообщений защищает две обменивающиеся сообщениями стороны от любой третьей, но не обеспечивает защиту каждой из сторон от другой. Поэтому в ситуациях, когда нет полного доверия между отправителем и получателем, требуется нечто большее, чем аутентификация. Наиболее привлекательное решение проблемы – использование электронной цифровой подписи (ЭЦП).

Цифровая подпись обеспечивает целый комплекс защиты, который было бы сложно осуществить любым другим способом. Она аналогична подписи, сделанной от руки.

ЭЦП должна обладать следующими свойствами:

1. Достоверность (ЭЦП должна давать возможность установить автора, дату и время подписи, а также достоверность содержимого сообщения на время подписи).
2. Неподдельность (Никто не может выдать себя за автора. ЭЦП доказывает, что именно подписавший и никто другой сознательно подписал документ).
3. Невозможность использовать подпись повторно (т.к. она – часть документа, два сообщения – две разных подписи).
4. Невозможность изменить подписанный документ (изменение документа – изменение подписи).
5. Невозможность отрицания авторства.

Из свойств следуют требования к ЭЦП:

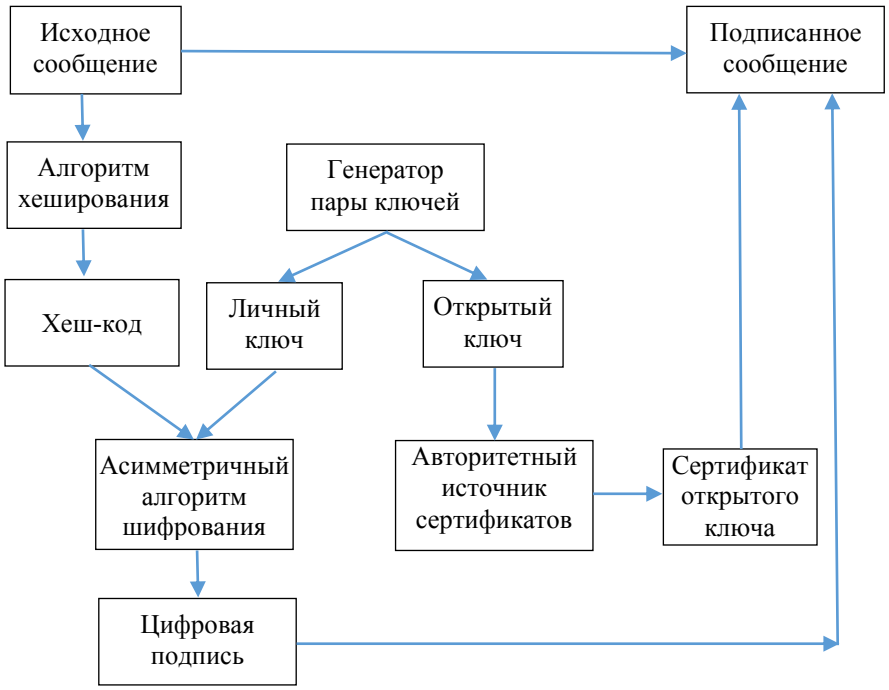
1. ЭЦП должна быть функцией подписываемого сообщения.
2. ЭЦП должна использовать информацию, уникальную для отправителя, чтобы предотвратить возможность фальсификации и отрицания авторства.
3. С точки зрения вычислений должно быть нереально фальсифицировать цифровую подпись (ни с помощью создания нового сообщения для имеющейся ЭЦП, ни с помощью создания фальшивой ЭЦП для имеющегося сообщения).
4. ЭЦП должна легко генерироваться.
5. ЭЦП должно быть легко распознать и проверить.
6. ЭЦП должно быть удобно хранить в запоминающих устройствах.

Всем этим требованиям удовлетворяет защищенная функция хеширования.

5. Создание и верификация цифровой подписи

Технология применения электронной цифровой подписипредполагает наличие сети абонентов, посылающих друг другу подписанные электронные документы. Для каждого абонента генерируется пара ключей: открытый и личный. Личный ключ хранится абонентом в тайне и используется для формирования ЭЦП.Открытый ключ известен всем другим пользователям и предназначен для проверки ЭЦП получателем документа.

Общая схема применения электронной цифровой подписи изображена на рисунке.



Берется исходное сообщение и создается его хеш-код при помощи одного из алгоритмов хеширования. Затем хеш-код шифруется при помощи личного ключа отправителя сообщения. Результат шифрования и называют электронной цифровой подписью

Никто, кроме владельца личного ключа, не сможет создать такую подпись, даже располагая оригиналом исходного сообщения. Никто не сможет изменить сообщение или создать поддельное сообщение так, чтобы обман не раскрылся.

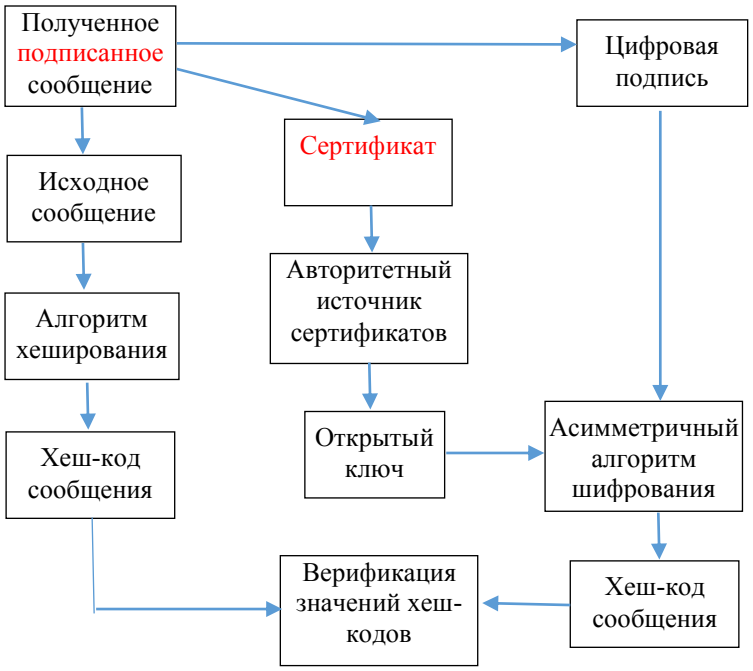
Подписанное сообщение формируется объединением исходного сообщения, его цифровой подписи и сертификата открытого ключа, соответствующего тому личному ключу, которым шифровался хеш.

При получении подписанное сообщение верифицируется. Получатель хочет убедиться в том, что сообщение действительно отправлено отправителем, а не кем-либо еще. Также получатель хочет убедиться, что сообщение не было изменено в пути следования.

Полученное сообщение разбивается на три компоненты – исходное сообщение, открытый ключ отправителя, цифровая подпись.

Получатель извлекает сертификат открытого ключа отправителя и проверяет его с помощью центра выдачи сертификатов. С помощью открытого ключа, содержащегося в сертификате, получатель может расшифровать хеш-код. После этого заново вычисляется хеш-код сообщения и сравнивается с полученным значением. Если хеши совпали, сообщение аутентично и не изменено.

Схема верификации ЭЦП приведена на рисунке.

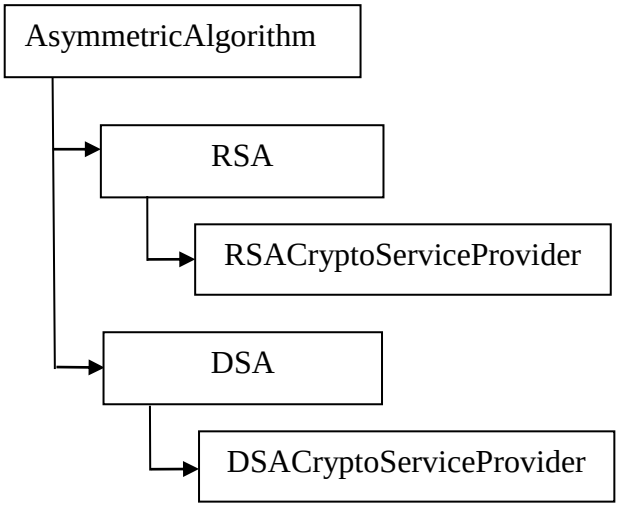


6. Класс AsymmetricAlgorithm

В библиотеке классов .NETSecurityподдерживаются асимметричные алгоритмы DSAи RSA. В основе всех асимметричных алгоритмов лежит класс AsymmetricAlgorithm.

Класс AsymmetricAlgorithmрасполагается в пространстве имен SystemSecurity.Cryptography и является абстрактным классом. Из него производятся классы алгоритмов: RSAи DSA, которые также являются абстрактными. Из классов RSAи DSA затем производятся классы RSACryptoServiceProvideri DSACryptoServiceProvider, которые обеспечивают реализацию алгоритмов. Эти классы являются оболочками для MicrosoftCrypto API.

Иерархия класса асимметричного алгоритма приведена на рисунке.



Способы работы с цифровой подписью с помощью классов DSACryptoServiceProvider и RSACryptoServiceProvider практически идентичны. Публичные методы и свойства классов DSACryptoServiceProvider и RSACryptoServiceProvidertакже во многом аналогичны.Конструктор класса автоматически генерирует ключевую информацию в момент создания экземпляра.

Некоторые публичные свойства классовпредставлены в таблице:

KeyExchangeAlgorithm	Свойство получает имя алгоритма обмена ключами
KeySize	Свойство получаетразмер ключа в битах
LegalKeySizes	Разрешенные размеры ключей
SignatureAlgorithm	Получает имя алгоритма ЭЦП

Основные публичные методы классов представлены в таблице:

CreateSignature	Создает цифровую подпись DSAдля заданного сообщения
VerifySignature	Верифицирует заданную подпись DSAдля заданного сообщения
SignData	Вычисляет хеш сообщения и подписывает его
VerifyData	Верифицирует заданную подпись, сравнивая ее с подписью, вычисленной для заданного сообщения
SignHash	Вычисляет подпись для заданного значения хеша
VerifyHash	Верифицирует заданную подпись, сравнивая ее с подписью, вычисленной для заданного хеша
ToXmlString	Создает и возвращает XML-представление текущего объекта
FromXmlString	Создает объект из XML-данных
ExportParameters	Экспортирует параметры DSAв объект DSAParameters
ImportParameters	Импортирует параметры DSAиз объектаDSAParameters

В функциях, обеспечиваемых этими методами, наблюдается некоторая избыточность, а это означает, что одну и ту же задачу можно решить разными способами.

7. Пример программы с использованием подписи RSAи алгоритма хеширования SHA-1

Создание и верификацию ЭЦП для заданного пользователем сообщения реализуем в виде отдельных методов CreateSignatureи VerifySignature.

МетодCreateSignature накладывает подпись на сообщение и состоит из следующих шагов.

Шаг 1. Преобразоватьведенное пользователем текстовое сообщение в массив байтов.

byte[] messagebytes = Encoding.UTF8.GetBytes(text);

Шаг 2. СоздатьобъектклассаSHA1CryptoServiceProvider.

SHA1sha1 = newSHA1CryptoServiceProvider();

Шаг 3. Сгенерировать хеш-код исходного сообщения.

byte[] hashbytes = sha1.ComputeHash(messagebytes);

Шаг 4. СоздатьобъектклассаRSACryptoServiceProviderс ключом по умолчанию.

RSACryptoServiceProviderrsa = newRSACryptoServiceProvider();

Шаг 5. Зашифровать хеш-кодличным ключом отправителя.

signaturebytes = rsa.SignHash(hashbytes, “1.3.14.3.2.26”);

Параметрами метода SignHashявляются: байтовый массив, содержащий сам хеш-код и идентификатор OID алгоритма хеширования SHA-1.

Шаг 6.Экспортироватьпараметры RSA при помощи метода ExportParametersс параметром false (извлекает открытый ключ, необходимый для верификации):

rsaparams = rsa.ExportParameters(false);

МетодVerifySignatureверифицирует подпись и состоит из следующих шагов.

Шаг 1 – Шаг 4 идентичны шагам метода CreateSignature.

Шаг 5. Вызвать метод ImportParameters с использованием того объекта rsaparams, что был создан ранее подписывающим методом. В этом случае параметры RSAбудут идентичны параметрам, использованным при создании подписи.

rsa.ImportParameters(rsaparams);

Шаг 6. Верифицировать подпись с помощью метода VerifyHashкласса RSACryptoServiceProvider:

bool match = rsa.VerifyHash(hashbytes, “1.3.14.3.2.26”, signaturebytes);

Для передачи информации между методами создания и верификации подписи используются три параметра:

1) text–строка, содержащая исходный текст сообщения.

2) RSAParametersrsaparams - объект, инкапсулирующий информацию открытого ключа.

3) byte[] signaturebytes – массив, содержащий цифровую подпись сообщения.

Задания на лабораторную работу

Написать программу создания и верификацииэлектронной цифровой подписи на основе указанных алгоритмов.

Вывести на экран значение хеш-кода в шестнадцатиричном формате, его размер, значение ЭЦП, параметры используемого асимметричного алгоритма (ключи, их размер и т.д.).

№ варианта	Асимметричный алгоритм	Хеш-алгоритм
1	RSA	MD5
2	RSA	SHA1
3	RSA	SHA256
4	RSA	SHA384
5	RSA	SHA512
6	DSA	MD5
7	DSA	SHA1
8	DSA	SHA256
9	DSA	SHA384
10	DSA	SHA512

Контрольные вопросы

1. Что такое хеш-код? Какие еще термины используются для обозначения данного понятия?
2. Какими свойствами должна обладать функция хеширования, чтобы ее можно было использовать в криптографии?
3. Какие современные алгоритмы хеширования вы знаете? Какие размеры дайджеста они поддерживают?
4. Какими свойствами должна обладать ЭЦП? Какие требования предъявляются к ЭЦП?
5. Какой метод распределения открытых ключей используется в схеме ЭЦП, предложенной в лабораторной работе?
6. Какие стандарты ЭЦП вы знаете? Какие размеры ключей и размеры ЭЦП в них используются?